03 | NumPy（下）：深度学习中的常用操作

2021-10-15 方远

《PyTorch深度学习实战》 课程介绍 >

讲述：方远
时长 17:07 大小 15.68M

你好，我是方远。

通过上节课的学习，我们已经对 NumPy 数组有了一定的了解，正所谓实践出真知，今天我们就以一个图像分类的项目为例，看看 NumPy 的在实际项目中都有哪些重要功能。

我们先从一个常见的工作场景出发，互联网教育推荐平台，每天都有千万量级的文字与图片的广告信息流入。为了给用户提供更加精准的推荐，你的老板交代你设计一个模型，让你把包含各个平台 Logo（比如包含极客时间 Logo）的图片自动找出来。

想要解决这个图片分类问题，我们可以分解成数据加载、训练与模型评估三部分（其实基本所有深度学习的项目都可以这样划分）。其中数据加载跟模型评估中，就经常会用到 NumPy 数组的相关操作。

那么我们先来看看数据的加载。

数据加载阶段

这个阶段我们要做的就是把训练数据读进来，然后给模型训练使用。训练数据不外乎这三种：图片、文本以及类似二维表那样的结构化数据。

不管使用 PyTorch 还是 TensorFlow，或者是传统机器学习的 scikit-learn，我们在读入数据这一块，都会先把数据转换成 NumPy 的数组，然后再进行后续的一系列操作。

对应到我们这个项目中，需要做的就是把训练集中的图片读入进来。对于图片的处理，我们一般会使用 Pillow 与 OpenCV 这两个模块。

虽然 Pillow 和 OpenCV 功能看上去都差不多，但还是有区别的。在 PyTorch 中，很多图片的操作都是基于 Pillow 的，所以当使用 PyTorch 编程出现问题，或者要思考、解决一些图片相关问题时，要从 Pillow 的角度出发。

下面我们先以单张图片为例，将极客时间的那张 Logo 图片分别用 Pillow 与 OpenCV 读入，然后转换为 NumPy 的数组。

Pillow 方式

首先，我们需要使用 Pillow 中的下述代码读入上面的图片。

复制代码

```
1 from PIL import Image
2 im = Image.open('jk.jpg')
```

```
3 im.size
4 输出：318，116
```

Pillow 是以二进制形式读入保存的，那怎么转为 NumPy 格式呢？这个并不难，我们只需要利用 NumPy 的 asarray 方法，就可以将 Pillow 的数据转换为 NumPy 的数组格式。

复制代码

```
1 import numpy as np
2
3 im_pillow = np.asarray(im)
4
5 im_pillow.shape
6 输出：(116, 318, 3)
```

OpenCV 方式：

OpenCV 的话，不再需要我们手动转格式，它直接读入图片后，就是以 NumPy 数组的形式来保存数据的，如下面的代码所示。

复制代码

```
1 import cv2
2 im_cv2 = cv2.imread('jk.jpg')
3 type(im_cv2)
4 输出：numpy.ndarray
5
6 im_cv2.shape
7 输出：(116, 318, 3)
```

结合代码输出可以发现，我们读入后的数组的最后一个维度是 3，这是因为图片的格式是 RGB 格式，表示有 R、G、B 三个通道。对于计算视觉任务来说，绝大多数处理的图片都是 RGB 格式，如果不是 RGB 格式的话，要记得事先转换成 RGB 格式。

这里有个地方需要你关注，Pillow 读入后通道的顺序就是 R、G、B，而 OpenCV 读入后顺序是 **B、G、R**。

模型训练时的通道顺序需与预测的通道顺序要保持一致。也就是说使用 Pillow 训练，使用 OpenCV 读入图片直接进行预测的话，不会报错，但结果会不正确，所以大家一定要注意。

接下来，我们就验证一下 Pillow 与 OpenCV 读入数据通道的顺序是否如此，借此引出有关 Numpy 数组索引与切片、合并等常见问题。

怎么验证这条结论呢？只需要将 R、G、B 三个通道的数据单独提取出来，然后令另外两个通道的数据全为 0 即可。

这里我给你说说为什么这样做。RGB 色彩模式是工业界的一种颜色标准，RGB 分别代表红、绿、蓝三个通道的颜色，将这三种颜色混合在一起，就形成了我们眼睛所能看到的所有颜色。

RGB 三个通道各有 256 个亮度，分别用数字 0 到 255 表示，数字越高代表亮度越强，数字 0 则是代表最弱的亮度。在我们的例子中，如果一个通道的数据再加另外两个全 0 的通道（相当于关闭另外两个通道），最终图像以红色格调（可以先看一下后文中的最终输出结果）呈现出来的话，我们就可以认为该通道的数据是来源于 R 通道，G 与 B 通道的证明同样可以如此。

好，首先我们提取出 RGB 三个通道的数据，这可以从数组的索引与切片说起。

索引与切片

如果你了解 Python，那么索引和切片的概念你应该不陌生。

就像图书目录里的索引，我们可以根据索引标注的页码快速找到需要的内容，而 Python

里的索引也是同样的功能，它用来定位数组中的某一个值。而切片意思就相当于提取图书中从某一页到某一页的内容。

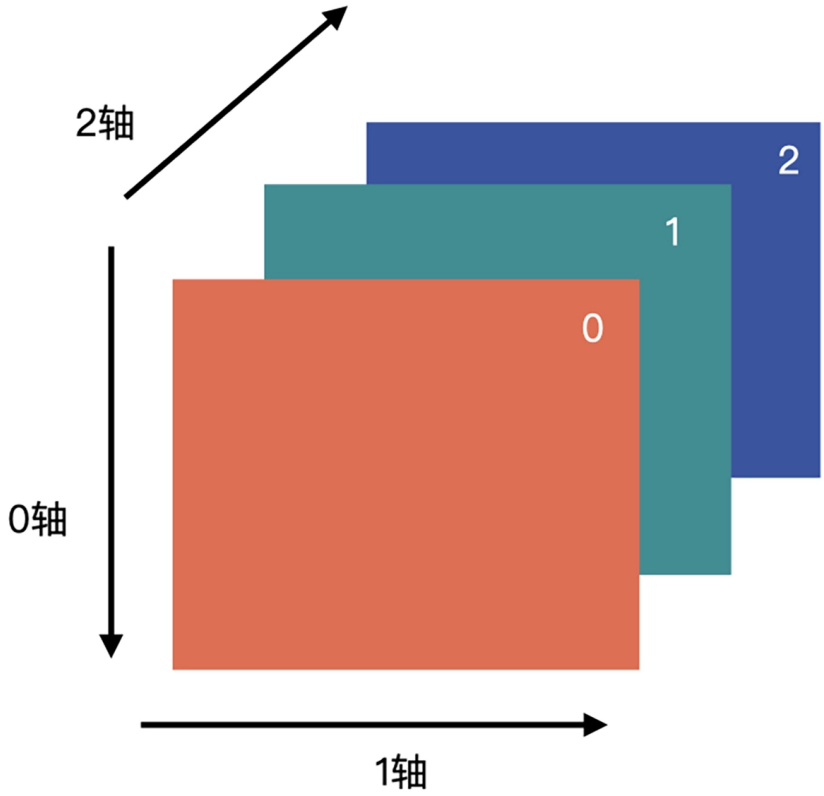
NumPy 数组的索引方式与 Python 的列表的索引方式相同，也同样支持切片索引。

这里需要你注意的是在 NumPy 数组中经常会出现用冒号来检索数据的形式，如下所示：

复制代码

```
1 im_pillow[:, :, 0]
```

这是什么意思呢？我们一起来看看。“：”代表全部选中的意思。我们的图片读入后，会以下图的状态保存在数组中。



极客时间

上述代码的含义就是取第三个维度索引为 0 的全部数据，换句话说就是，取图片第 0 个通道的所有数据。

这样的话，通过下面的代码，我们就可以获得每个通道的数据了。

复制代码

```
im_pillow_c1 = im_pillow[:, :, 0]
im_pillow_c2 = im_pillow[:, :, 1]
im_pillow_c3 = im_pillow[:, :, 2]
```

获得了每个通道的数据，接下来就需要生成一个 3 维全 0 数组，全 0 数组的形状除了最后一维为 2，其余两维要与 im_pillow 的形状相同。

全 0 数组你还记得怎么生成吗？可以自己先思考一下，生成的代码如下所示。

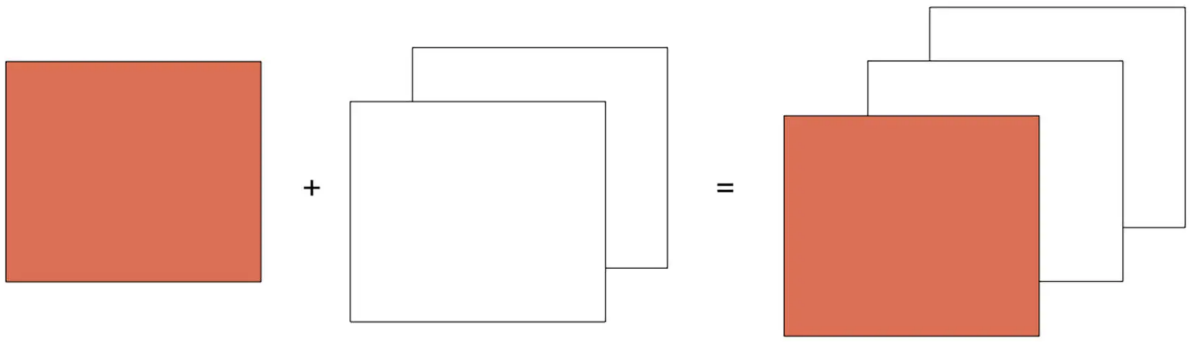
复制代码

```
zeros = np.zeros((im_pillow.shape[0], im_pillow.shape[1], 2))
zeros.shape
输出：(116, 318, 2)
```

然后，我们只需要将全 0 的数组与 im_pillow_c1、im_pillow_c2、im_pillow_c3 进行拼接，就可以获得对应通道的图像数据了。

数组的拼接

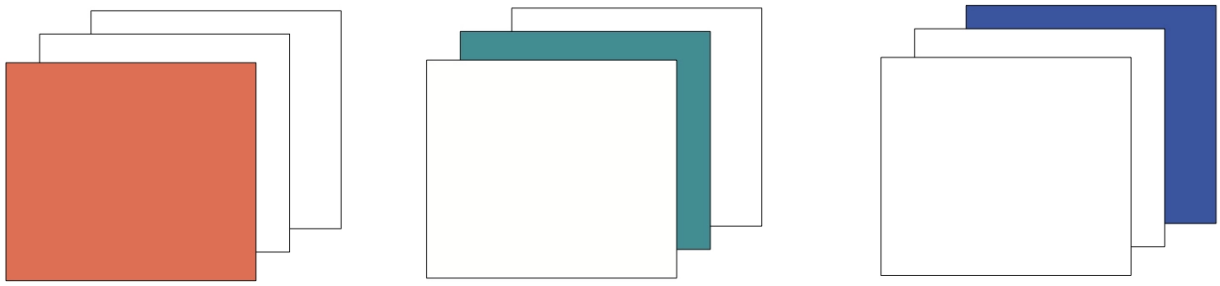
刚才我们拿到了单独通道的数据，接下来就需要把一个分离出来的数据跟一个全 0 数组拼接起来。如下图所示，红色的可以看作单通道数据，白色的为全 0 数据。



极客时间

NumPy 数组为我们提供了 np.concatenate((a1, a2, ...), axis=0) 方法进行数组拼接。其中，a1，a2, ...就是我们要合并的数组；axis 是我们要沿着哪一个维度进行合并，默认是沿着 0 轴方向。

对于我们的问题，是要沿着 2 轴的方向进行合并，也是我们最终的目标是要获得下面的三幅图像。



极客时间

那么，我们先将 im_pillow_c1 与全 0 数组进行合并，生成上图中最左侧的数组，有了图像的数组才能获得最终图像。合并的代码跟输出结果如下：

复制代码

```
im_pillow_c1_3ch = np.concatenate((im_pillow_c1, zeros), axis=2)
---------------------------------------------------------------------------
AxisError                                 Traceback (most recent call last)
<ipython-input-65-90bba90337ff> in <module>
----> 1 im_pillow_c1_3ch = np.concatenate((im_pillow_c1, zeros), axis=2)
<__array_function__ internals> in concatenate(*args, **kwargs)
AxisError: axis 2 is out of bounds for array of dimension 2
```

看到这里你可能很惊讶，竟然报错了？错误的原因是在 2 维数组中，axis 如果等于 2 的话会越界。

我们看看 im_pillow_c1 与 zeros 的形状。

复制代码

```
im_pillow_c1.shape
输出：(116, 318)
zeros.shape
输出：(116, 318, 2)
```

原来是我们要合并的两个数组维度不一样啊。那么如何统一维度呢？将 im_pillow_c1 变成 (116, 318, 1) 即可。

方法一：使用 np.newaxis

我们可以使用 np.newaxis 让数组增加一个维度，使用方式如下。

复制代码

```
im_pillow_c1 = im_pillow_c1[:, :, np.newaxis]
im_pillow_c1.shape
输出：(116, 318, 1)
```

运行上面的代码，就可以将 2 个维度的数组转换为 3 个维度的数组了。

这个操作在你看深度学习相关代码的时候经常会看到，只不过 PyTorch 中的函数名 unsqueeze(), TensorFlow 的话是与 NumPy 有相同的名字，直接使用 tf.newaxis 就可以了。

然后我们再次将 im_pillow_c1 与 zeros 进行合并，这时就不会报错了，代码如下所示：

复制代码

```
im_pillow_c1_3ch = np.concatenate((im_pillow_c1, zeros), axis=2)
im_pillow_c1_3ch.shape
输出：(116, 318, 3)
```

方法二：直接赋值

增加维度的第二个方法就是直接赋值，其实我们完全可以生成一个与 im_pillow 形状完全一样的全 0 数组，然后将每个通道的数值赋值为 im_pillow_c1、im_pillow_c2 与 im_pillow_c3 就可以了。我们用这种方式生成上图中的中间与右边图像的数组。

复制代码

```
im_pillow_c2_3ch = np.zeros(im_pillow.shape)
im_pillow_c2_3ch[:,:,1] = im_pillow_c2[:,:,0]

im_pillow_c3_3ch = np.zeros(im_pillow.shape)
im_pillow_c3_3ch[:,:,2] = im_pillow_c3[:,:,0]
```

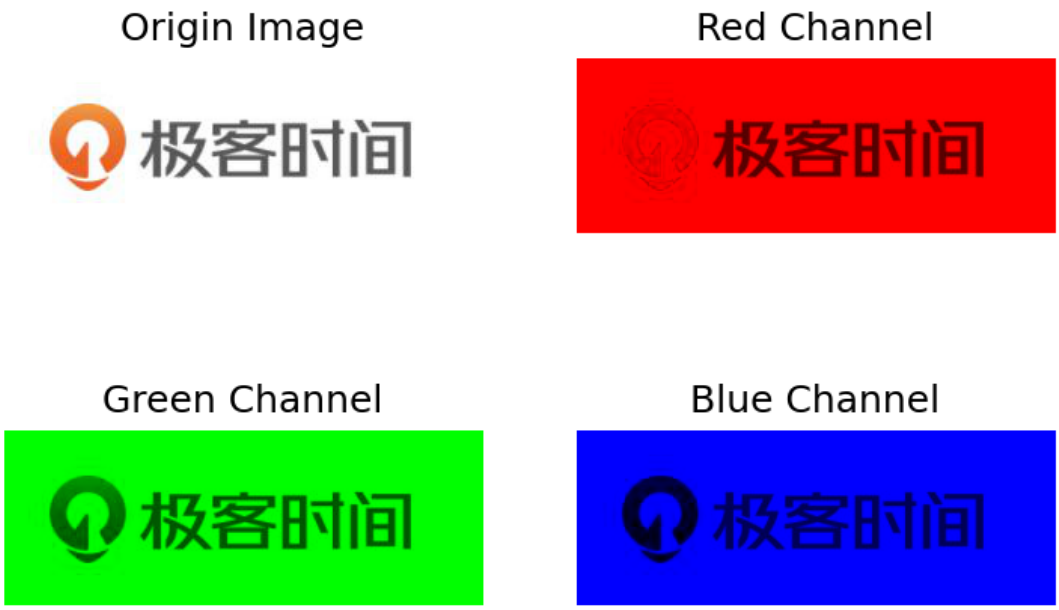
这样的话，我们就可以将三个通道的 RGB 图片打印出来了。

关于绘图，你可以使用 matplotlib 进行绘图，它是 NumPy 的绘图库。如果你需要绘图，可以在这个网站上找到各种各样的例子，然后根据它提供的代码进行修改，具体如何绘图我就不展开了。

说回我们的通道顺序验证问题，完成前面的操作后，你可以用下面的代码将原图、R 通道、G 通道与 B 通道的 4 幅图打印出来，你看是不是 RGB 顺序的呢？

复制代码

```
from matplotlib import pyplot as plt
plt.subplot(2, 2, 1)
plt.title('Origin Image')
plt.imshow(im_pillow)
plt.axis('off')
plt.subplot(2, 2, 2)
plt.title('Red Channel')
plt.imshow(im_pillow_c1_3ch.astype(np.uint8))
plt.axis('off')
plt.subplot(2, 2, 3)
plt.title('Green Channel')
plt.imshow(im_pillow_c2_3ch.astype(np.uint8))
plt.axis('off')
plt.subplot(2, 2, 4)
plt.title('Blue Channel')
plt.imshow(im_pillow_c3_3ch.astype(np.uint8))
plt.axis('off')
plt.savefig('./rgb_pillow.png', dpi=150)
```



深拷贝（副本）与浅拷贝（视图）

刚才我们通过获取图片通道数据的练习，不过操作确实比较繁琐，介绍这些方法也主要是为了让你掌握切片索引和数组拼接的知识点。

其实我们还有一种更加简单的方式获得三个通道的 BGR 数据，只需要将图片读入后，直接将其中的两个通道赋值为 0 即可。代码如下所示：

复制代码

```
from PIL import Image
import numpy as np

im = Image.open('jk.jpg')
im_pillow = np.asarray(im)
im_pillow[:,:,1:]=0
输出：
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-146-789bda58f667> in <module>
      4 im = Image.open('jk.jpg')
      5 im_pillow = np.asarray(im)
----> 6 im_pillow[:,:,1:-1]=0
ValueError: assignment destination is read-only
```

运行刚才的代码，报错提示说数组是只读数组，没办法进行修改。那怎么办呢？我们可以使用 copy 来复制一个数组。

说到 copy() 的话，就要说到浅拷贝与深拷贝的概念，上节课我们说到创建数组时就提过，np.array() 属于深拷贝，np.asarray() 则是浅拷贝。

简单来说，浅拷贝或称视图，指的是与原数组共享**数据**的数组，请注意，只是数据，没有说共享形状。视图我们通常使用 view() 来创建。常见的切片操作也会返回对原数组的浅拷贝。

请看下面的代码，数组 a 与 b 的数据是相同的，形状确实不同，但是修改 b 中的数据后，a 的数据同样会发生变化。

复制代码

```
a = np.arange(6)
print(a.shape)
输出：(6,)
print(a)
输出：[0 1 2 3 4 5]

b = a.view()
print(b.shape)
输出：(6,)
b.shape = 2, 3
print(b)
输出：[[0 1 2]
 [3 4 5]]
b[0,0] = 111
print(a)
输出：[111   1   2   3   4   5]
print(b)
输出：[[111   1   2]
 [  3   4   5]]
```

而深拷贝又称副本，也就是完全复制原有数组，创建一个新的数组，修改新的数组不会影响原数组。深拷贝使用 copy() 方法。

所以，我们将刚才报错的程序修改成下面的形式就可以了。

```
im_pillow = np.array(im)
im_pillow[:,:,1:]=0
```

复制代码

可别小看深拷贝和浅拷贝的区别。这里讲一个我以前遇到的坑吧，我曾经要开发一个部署在手机端的人像分割模型。

为了提高模型的分割效果，我考虑了新的实验方法——将前一帧的数据也作为当前帧的输入进行考虑，训练阶段没有发生问题，但是在调试阶段发现模型的效果非常差。

后来经过研究，我才发现了问题的原因。原因是我为了可视化分割效果，我将前一帧的数据进行变换打印出来。同时，我错误的采用了浅拷贝的方式，将前一帧的数据传入当前帧，所以说传入到当前帧的数据是经过变化的，而不是原始的输出。

这时再传入当前帧，自然无法得到正确结果。当时因为这个坑，差点产生要放弃这个实验的想法，后面改成深拷贝才解决了问题。

好了，讲到这里，你是否可以用上述的方法对 OpenCV 读取图片读入通道顺序进行一下验证呢？

模型评估

在模型评估时，我们一般会将模型的输出转换为对应的标签。

假设现在我们的问题是将图片分为 2 个类别，包含极客时间的图片与不包含的图片。模型会输出形状为 (2,) 的数组，我们把它叫做 probs，它存储了两个概率，我们假设索引为 0 的概率是包含极客时间图片的概率，另一个是其它图片的概率，它们两个概率的和为 1。如果极客时间对应的概率大，则可以推断该图片为包含极客时间的图片，否则为其他图片。

简单的做法就是判断 probs[0]是否大于 0.5，如果大于 0.5，则可以认为图片是我们要寻找的。

这种方法固然可以，但是如果我们需要判断图片的类别有很多很多种呢？

例如，有 1000 个类别的 ImageNet。也许你会想到遍历这个数组，求出最大值对应的索引。

那如果老板让你找出概率最大的前 5 个类别呢？有没有更简单点的方法？我们继续往下看。

Argmax Vs Argmin：求最大 / 最小值对应的索引

NumPy 的 argmax(a, axis=None) 方法可以为我们解决求最大值索引的问题。如果不指定 axis，则将数组默认为 1 维。

对于我们的问题，使用下述代码即可获得拥有最大概率值的图片。

```
1 np.argmax(probs)
```

复制代码

Argmin 的用法跟 Argmax 差不多，不过它的作用是获得具有最小值的索引。

Argsort：数组排序后返回原数组的索引

那现在我们再把问题升级一下，比如需要你将图片分成 10 个类别，要找到具有最大概率的前三个类别。

模型输出的概率如下：

```
1 probs = np.array([0.075, 0.15, 0.075, 0.15, 0.0, 0.05, 0.05, 0.2, 0.25])
```

复制代码

这时，我们就可以借助 argsort(a, axis=-1, kind=None) 函数来解决该问题。np.argsort 的作用是对原数组进行从小到大的排序，返回的是对应元素在原数组中的索引。

np.argsort 包括后面这几个关键参数：

a 是要进行排序的原数组；

axis 是要沿着哪一个轴进行排序，默认是 -1，也就是最后一个轴；

kind 是采用什么算法进行排序，默认是快速排序，还有其他排序算法，具体你可以看看数据结构的排序算法。

我们还是结合例子来理解，你可以看看下面的代码，它描述了我们使用 argsort 对 probs 进行排序，然后返回对应坐标的全过程。

复制代码

```
probs_idx_sort = np.argsort(-probs)  #注意，加了负号，是按降序排序
probs_idx_sort
输出：array([8, 7, 1, 3, 0, 2, 5, 6, 4])
#概率最大的前三个值的坐标
probs_idx_sort[:3]
输出：array([8, 7, 1])
```

小结

恭喜你，完成了这一节课的学习。这一节介绍了一些常用且重要的功能。几乎在所有深度学习相关的项目中，你都会常常用到这些函数，当你阅读别人的代码的时候也会经常看到。

让我们一起来复习一下今天学到的这些函数，我画了一张表格，给你总结了它们各自的关键功能和使用要点。

函数名	关键功能	使用要点/注意事项
view()	浅拷贝/视图，用来获取与原数组共享数据的数组	只共享数据，无法共享形状
copy()	深拷贝/副本，创建一个完全一模一样的新数组	不共享数据，修改新数组不会影响原数组
concatenate()	多个数组拼接	需要注意沿哪个轴拼接
argmax()	返回具有最大值的索引	/
argsort()	对原数组排序，返回对应的索引	/

极客时间

我觉得 NumPy 最难懂的还是上节课的轴，如果你把轴的概念理解清楚之后，理解今天的内容会更加轻松。理解了原理之后，关键还是动手练习。

每课一练

给定数组 scores，形状为（256，256，2），scores[: , :, 0] 与 scores[:, :, 1]对应位置元素的和为 1，现在我们要根据 scores 生产数组 mask，要求 scores 通道 0 的值如果大于通道 1 的值，则 mask 对应的位置为 0，否则为 1。

scores 如下，你可以试试用代码实现：

复制代码

```
scores = np.random.rand(256, 256, 2)
scores[:,:,1] = 1 - scores[:,:,0]
```

欢迎你在留言区记录你的疑问或者收获，也推荐你把这节课分享给你的朋友。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 0 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 02 | NumPy（上）：核心数据结构详解

精选留言 (5)

写留言

文森特没有眼泪

2021-10-15

```
mask = (s[:,:,0] > s[:,:,1])
mask = mask.astype('int')
```

作者回复: 你好，文森特没有眼泪。首先，谢谢你的留言，然后为你的思路点赞👍。
不过稍微有点小瑕疵，题目要求的是，如果通道0的值大于通道1的值，那么mask对应位置的数值就为0，否则为1。
你这样做的结果正好是相反了。
另外，你可以考虑考虑如何使用聚合函数np.argmax来求解，思考一下它的axis应该等于几，才能获得正确的答案。^^

1

IUniverse
2021-10-15

import numpy as np

scores=np.random.rand(256,256,2)
scores[:,:,0]=1-scores[:,:,1]
mask=np.argmax(scores,axis=2)…
展开

作者回复: 你好，IUniverse。感谢你的留言。你的回答是正确的👍。^^

Yuhan
2021-10-15

方法一：
result = np.argmax(scores, axis=2)
方法二：
result = (scores[:, :, 0] < scores[:, :, 1]).astype('int')
展开

作者回复: 你好，Yuhan。感谢你的留言。👍，你的答案是正确的。^^

莎莎 🍪
2021-10-15

https://matplotlib.org/stable/gallery/index.html
展开

lwg0452

2021-10-15

```
import numpy as np

scores = np.random.rand(256, 256, 2)
scores[:, :, 1] = 1 - scores[:, :, 0]
print(scores)…
```

展开

作者回复: 你好，lwg0452。感谢你的留言，首先为你的思路点赞👍。

不过稍微有点小瑕疵，题目要求的是，如果通道0的值大于通道1的值，那么mask对应位置的值就是0，否则为1。

你这样做的结果正好是相反了。

另外，你可以考虑考虑如何使用聚合函数np.argmax来求解，思考一下它的axis应该等于几，才能获得正确的答案。：）